UF Research Computing

Introduction to Perl

Matt Gitzendanner

Oleksandr Moskalenko

Perl

- General purpose, interpreted language
- Developed in 1987 for processing text files
- Has many uses, but very popular in bioinformatics applications
  - Sequence data are text

Many Useful Functions

```
substr($line,5,10);
length($line);
reverse($DNA);
$seq=~ s/T/U/g;
$seq=~ tr/ATCG/TAGC/;
@bits=split(/ /, $line);
if($name=~/^>(.*)/)
```

Perl is great with Regular Expressions

Variables

- Name that holds a value
- In Perl: `$variable`
- Type: integer, float, boolean, string, etc.: Perl determines from context

Strings, numbers, math

```
$DNA1='AGTGACGTGACGTACGACTGATCA';
$DNA2='GTAGATCA';
$nt1=length($DNA1);
$nt2=length($DNA2);

$DNA3= $DNA1 . $DNA2;
$nt3=$nt1+$nt2;

print "$DNA3\n";
print $nt1 . "+" . $nt2 . "=" . length($DNA3) . "=" . $nt3 . "\n";
```

```
AGTGACGTGACGTACGACTGATCAGTAGATCA
24+8=32=32
```

Arrays

```
#!/usr/bin/perl

@nucleotides=("G","A","T","C");

print "$nucleotides[1] \n";
```

```
A
```

@nucleotides[	G	0
	A	1
	T	2
	C]	3

index

Hashes

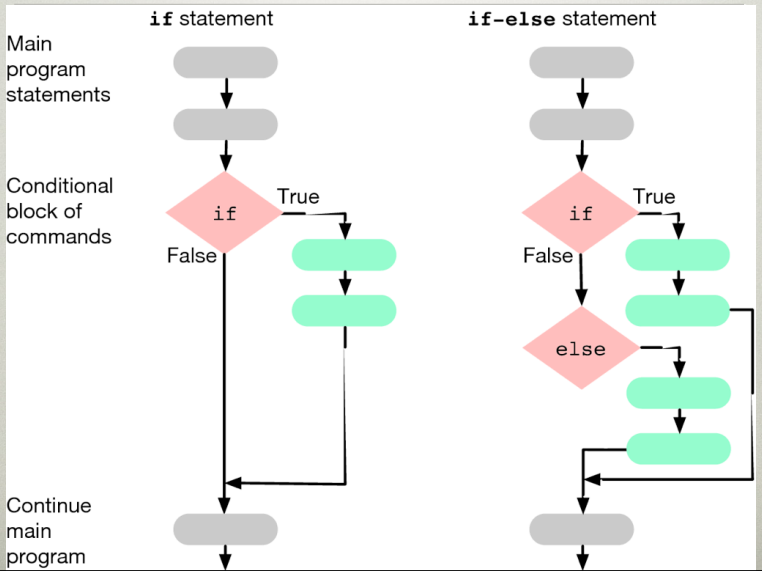
```
%gene_expression=(
     'adh'=> 4.5,
     'g3dp' => 6.2,
     'ndh' => 3.2,
     );

print $gene_expression{ndh} . "\n";
```

key => value

```
3.2
```

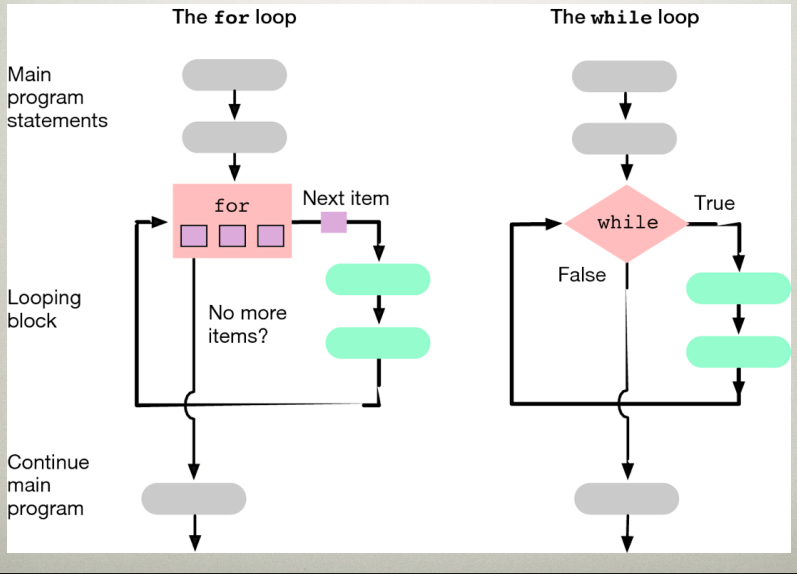
if and if-else



if, elsif, else

```
$nt =substr($line,$k,1);
if ($nt eq "g")
    {$g++;}
elsif ($nt eq "a")
    {$a++;}
elsif ($nt eq "t")
    {$t++;}
elsif ($nt eq "c")
    {$c++;}
else
    {print $nt . "?\n";}
```

for and while loops



for loop

```
for ($k=0; $k<$i; $k++)
{
    $nt =substr($line,$k,1);
    if ($nt eq "g")
        {$g++;}
    elsif ($nt eq "a")
        {$a++;}
    elsif ($nt eq "t")
        {$t++;}
    elsif ($nt eq "c")
        {$c++;}
    else
        {print $nt . "?\n";}
}
```

foreach

```
@files= glob "*.fa";

foreach $file(@files) {
    $seqs=`cat $file | grep -c \\>`;
    chomp($seqs);
    print $file ." has $seqs sequences in it.\n";
}
```

Shell commands executed with back-tick : `` `command` ``

```
BNTL-18S_1.fa has 107054 sequences in it.
BNTL-18S_2.fa has 108338 sequences in it.
OSIP.assembly.fa has 142475 sequences in it.
OSIP-reads_1.fa has 8670493 sequences in it.
```

File input/output

```
for my $file(@files) {

    open FH, "<$file" or die "Could not open $file: $!";
    while (<FH>)
    {
      Do something here
    }
    close FH;
}
```

< Read from a file

File input/output

```
@files= glob "*.fa";
open OUT, ">Num_seqs.txt" or die "Could not open Num_seqs.txt: $!";

foreach $file(@files) {
    $seqs=`cat $file | grep -c \\>`;
    chomp($seqs);
    print OUT $file ." has $seqs sequences in it.\n";
}
```

> Write to a file
>> Append to a file

BioPerl

- ✦Installed on UF HPC cluster
  - `module load perl`
- ✦For some things very good
  - E.g.: parsing blast output
- ✦For some things can fail
  - E.g.: reading fastq files (@ quality scores)

BioPerl Blast Parsing

```
$in = new Bio::SearchIO(-format => 'blast',
                        -file   => $file);
while( $result = $in->next_result ) {
  $hit=$result->next_hit;
 if (! defined ($hit)) { next;}
 $hsp=$hit->next_hsp;

    if ( ($hit->description=~ m/chloropl/) or ($hit->description=~ m/mitochond/) ) {

        my ($query_start, $query_end) = $hsp->range('query');
        my ($hit_start, $hit_end) = $hsp->range('hit');
        print OUTFILE $sequence_number, "\t",
                $result->query_name, "\t",
                $result->query_length, "\t",
                $result->num_hits, "\t",
                $hit->description, "\t",
                $hsp->hsp_length, "\t",
                $query_start, "-", $query_end,"\t",
                $hit_start, "-", $hit_end,"\n";

      $sequence_number++;
    }
}
```

Object Oriented Programming

- Most modern programming languages include OOP
  - Perl: POOP
- Object is the thing that provides access to data
- Class is description of attributes of a particular kind of object and methods for using data in the object

SearchIO POOP

```
my $in = new Bio::SearchIO(-format => 'blast',
                           -file   => $file);
```

Use the new method of class SearchIO to create an object with the format of blast and file in $file

```
while( $result = $in->next_result ) {
  $hit=$result->next_hit;
```

Use the next_result method of class SearchIO to look at each blast result of object $in

Use the next_hit method of SearchIO to look at each hit

Stealing code is…easier than writing it

- Find a script that is similar
  - Google is your friend…

- Perl sites like:
  - http://www.perl.org/docs.html
  - http://www.tizag.com/perlT/

How to get help

- Asking for help
  - Support Request Tickets
    - http://support.hpc.ufl.edu
    - Use for everything - not just software bugs but for any questions or help requests
    - Searchable database of solutions
  - When you don't have access to web
    - support@hpc.ufl.edu
    - om@hpc.ufl.edu (Biological Support)
    - magitz@ufl.edu (Bio training and Q/A)

Documentation

- UF HPC Encyclopedia
  - http://wiki.hpc.ufl.edu
    - Documents on hardware and software resources
    - User guides
    - Sample submission scripts
    - Research-specific sections
  - http://hpc.ufl.edu/support
    - Frequently Asked Questions
    - Account set up and maintenance

Training Schedule

- ✓ Jan 26: Intro to UFHPC, getting started
- ✓ Feb 2: Galaxy overview and basic usage
- ✓ Feb 9: Intro to Linux shell
- ✓ Feb 16: No session
- ✓ Feb 23: Modules and basic submission scripts
- ✓ Mar 1: Working with NGS data
- ✓ Mar 8: No session – Spring break
- ✓ Mar 15: Perl intro
- Mar 22: Perl in practice with Gordon Burleigh
- Mar 29: Statistical analyses at HPC intro
- Apr 5: R and SAS in practice with TBA
- Apr 12: TBA
- Apr 19: TBA


Thank
you!